University of Pittsburgh

Algorithms and Data Structures 2
CS 1501

Spring 2022

Sherif Khattab

ksm73@pitt.edu

(Slides are adapted from Dr. Ramirez's and Dr. Farnan's CS1501 slides.)

Announcements

- Upcoming deadlines:
  - Homework 4 is due on 2/14
  - Lab 4 is due on 2/18
  - Homework 5 is due on 2/21

Previous lecture …

- Binary Tree Traversal
  - Pre-order
  - In-order
  - Post-order
  - Level order
  - Searching in a BST
- Recursive and iterative

CourseMIRROR Reflections: Most Interesting

- I found the different traversal interesting instead of just a top to bottom concept.
- Giving candies
- The stack implementation was interesting and how it compares to the recursion
- The contour paths of the different traversal types. Very helpful for remember the differences!
- Ways of traversing a BST
- Relating code for traversal with a visual description of what's happening
- how inorder traversal returns the tree in a sorted order

CourseMIRROR Reflections: Most Confusing

- how the stack works in relation to the traversals. The traversals make sense but the order of push and pop does not.
- How the call stack works with traversing (it felt like we went through the material kind of fast)
- if you could provide typed Java code for iterative traversal instead of handwritten pseudocode that would make it more clear to understand
- The post order traversing! I am confused on the place to push, is all of the x will be in the visit listing for next second?
- Developing iterative code from recursive code; how to get from recursive to iterative
- I would like to go over DFS and BFS more

Prefix Searching Problem

- Input:
  - a (large) dynamic set of data items in the form of
    - *n* (key, value) pairs; key is a string from an alphabet of size *R*
    - Each key has *b* bits or *w* characters (the chars are from the alphabet)
    - What is the relationship between *b* and *w*?
  - a target *string*
- Output:
  - 0: string is not a prefix of nor equal to any of the keys
  - 1: string is a prefix of at least one key but not equal to any
  - 2: string is not a prefix of any key but is equal to a key
  - 3: string is both a prefix of at least one key and equal to one of the keys

Let's create an ADT!

- The Prefix Symbol Table ADT
  - A set of (key, value) pairs
- Operations of the PST ADT
  - insert
  - delete
  - prefixSearch
  - search

Prefix Symbol Table Implementations

- Array
  - Unsorted
  - Sorted
- Linked List
  - Unsorted
  - Sorted
- Binary Search Tree
- Red-Black Binary Search Tree

Digital Search Trees (DSTs)

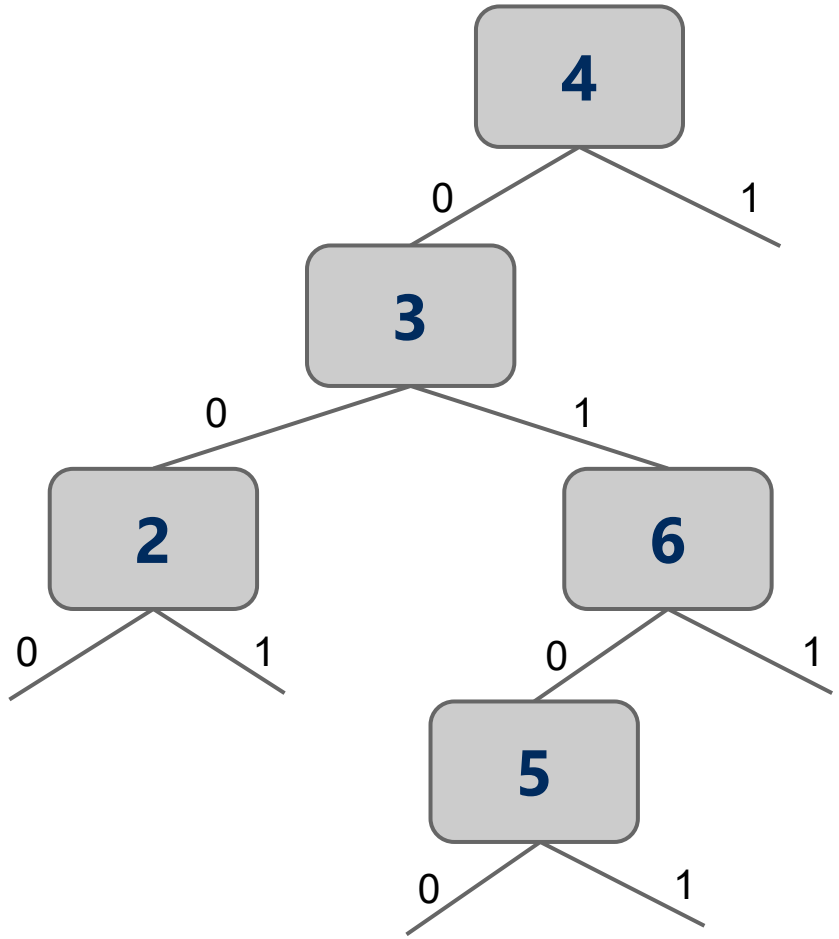
Instead of looking at less than/greater than, lets go left right based on the bits of the key, so we again have 4 options:

- Node ref is null, k not found
- k is equal to the current node's key, k is found
- current bit of k is 0, continue to left child
- current bit of k is 1, continue to right child

DST example

Insert:

4	0100
3	0011
2	0010
6	0110
5	0101



Search:

3	0011
7	0111

Analysis of digital search trees

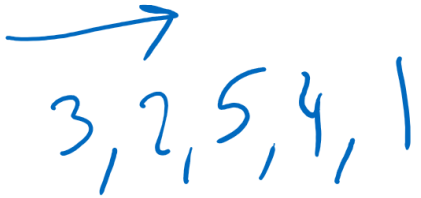
- Runtime?

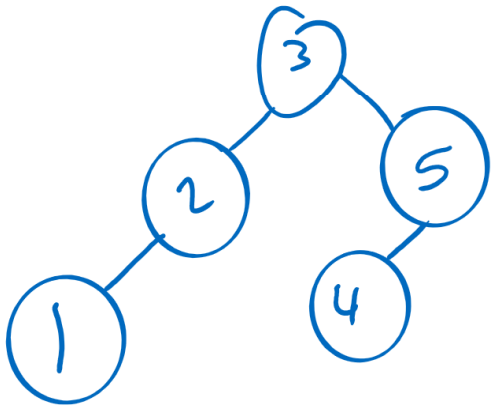
- We end up doing many comparisons against the full key, can we improve on this?

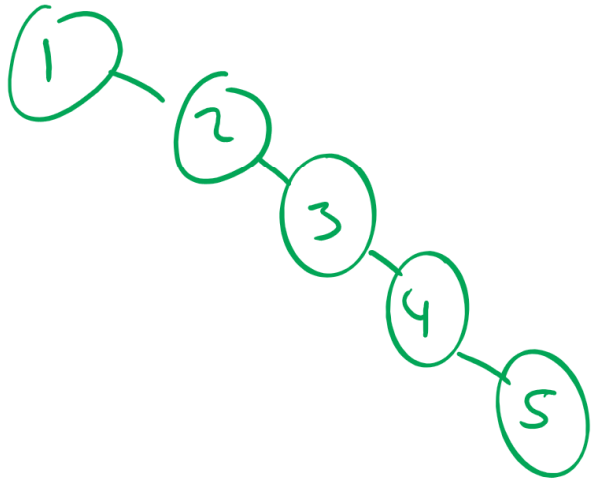
Average-case vs. Worst-case

$$\text{average case runtime} = \sum_{\text{all cases}} P(\text{case}_i) \times \text{runtime for case}_i$$

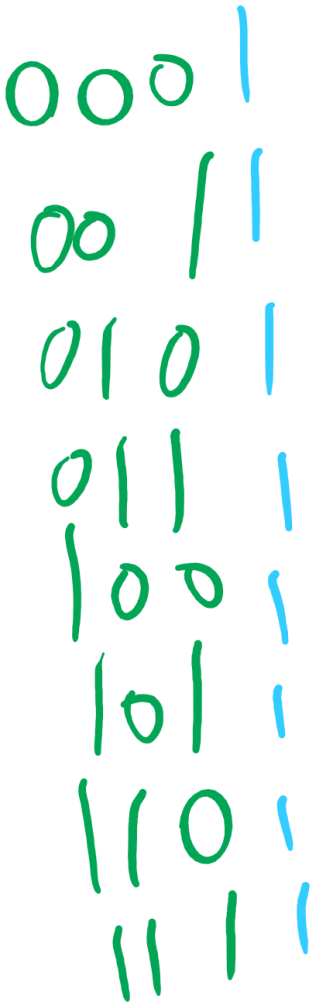
$$P(\text{case}_i) = \frac{1}{\#\text{ cases}}$$







n vs. bit length



min # bits $= \lceil \log_2 n \rceil$

$b \geq \lceil \log_2 n \rceil$

Let's create an ADT!

- The Prefix Symbol Table ADT
  - A set of (key, value) pairs
- Operations of the PST ADT
  - insert
  - delete
  - prefixSearch
  - search

Radix search tries (RSTs)

- Trie as in retrieve, pronounced the same as "try"
- Instead of storing keys as nodes in the tree, we store them implicitly as paths down the tree
  - Interior nodes of the tree only serve to direct us according to the bitstring of the key
  - Values can then be stored at the end of key's bit string path

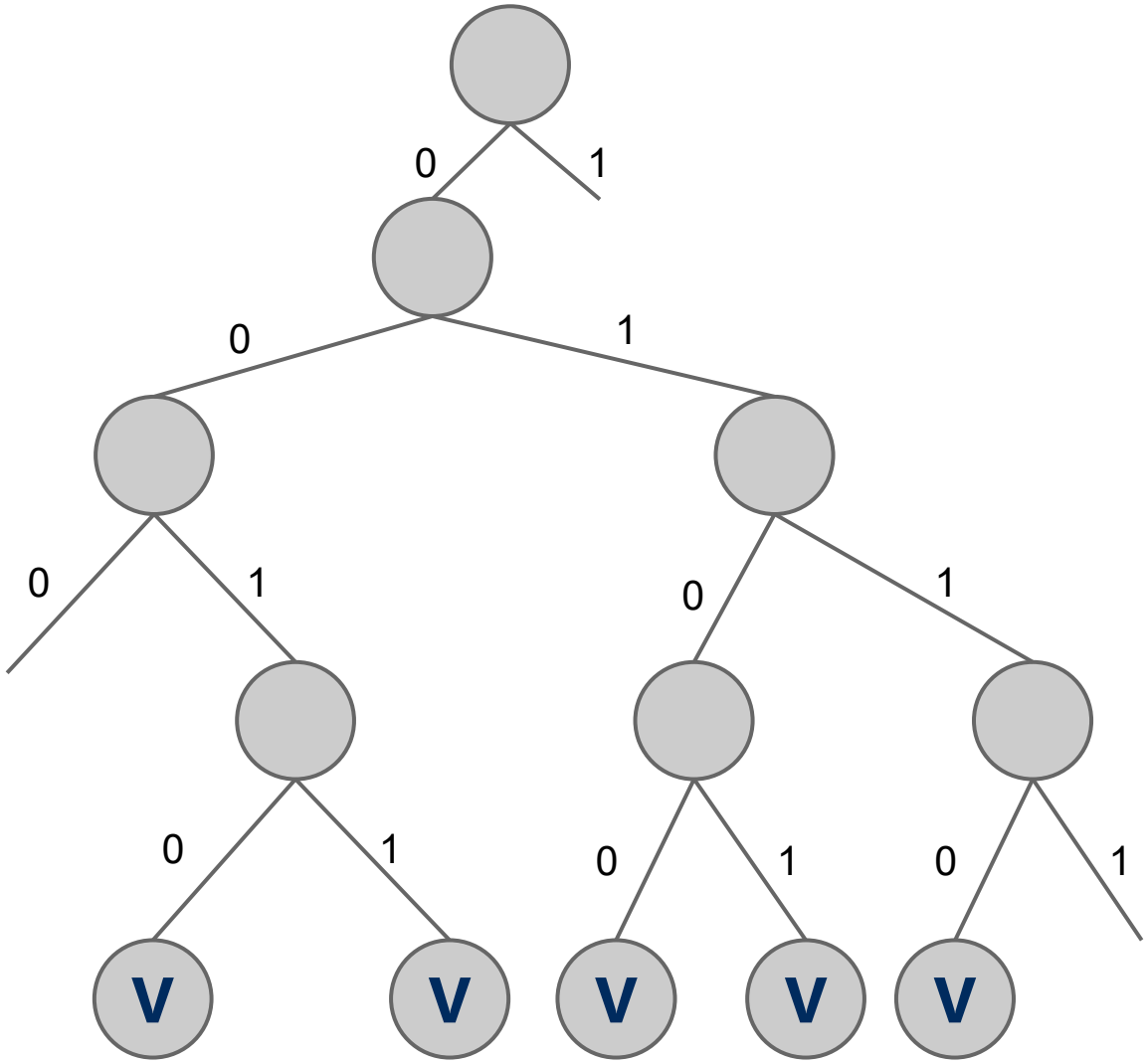
RST example

Insert:

4	0100
3	0011
2	0010
6	0110
5	0101

Search:

3	0011
7	0111



RST analysis

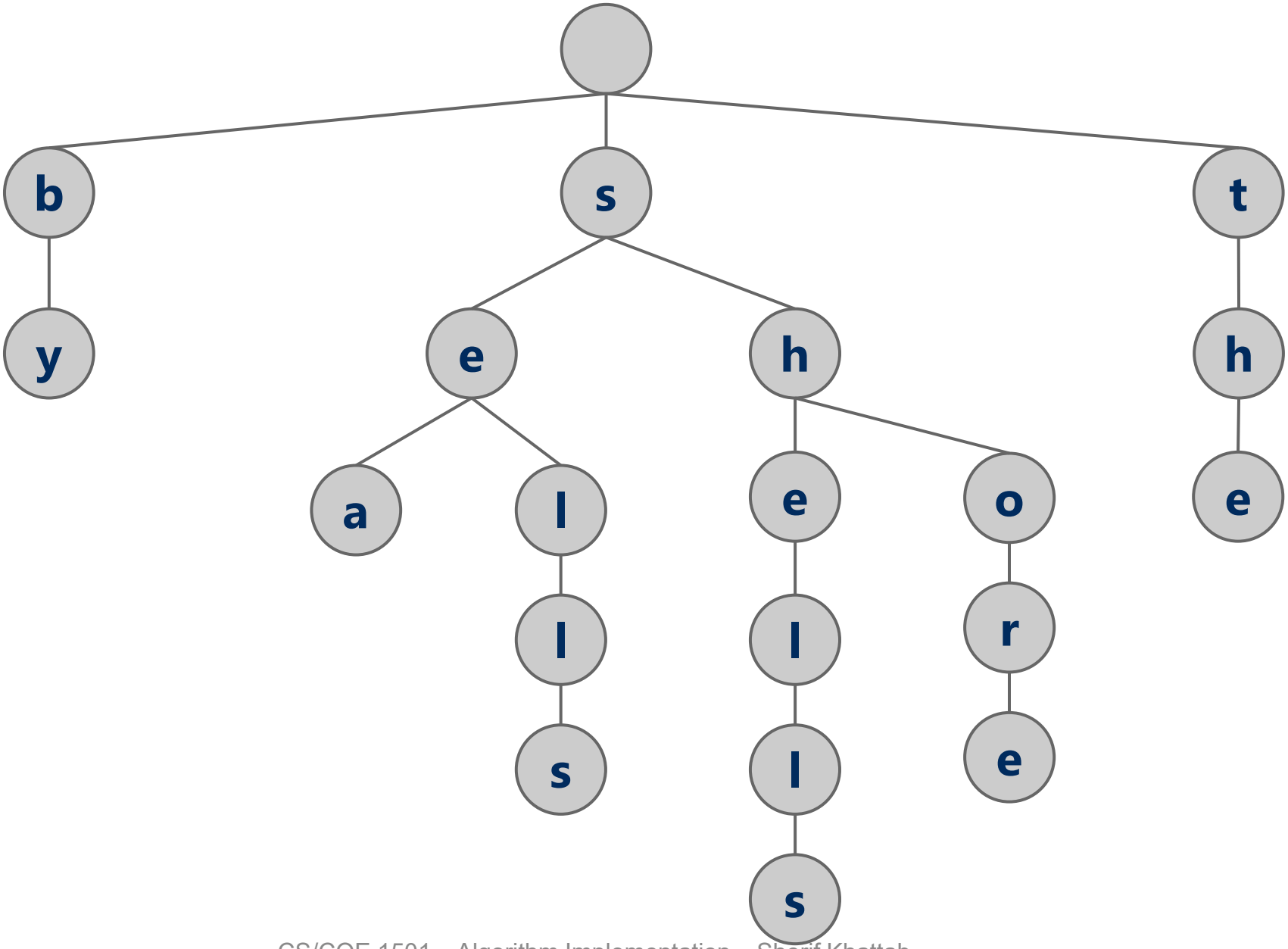
- Runtime?

- Would this structure work as well for other key data types?
  - Characters?
  - Strings?

Larger branching factor tries

- In our binary-based Radix search trie, we considered one bit at a time
- What if we applied the same method to characters in a string?
  - What would this new structure look like?
- Let's try inserting the following strings into an trie:
  - she, sells, sea, shells, by, the, sea, shore

Another trie example



Please submit your reflections by using the CourseMIRROR App

If you are having a problem with CourseMIRROR, please send an email to coursemirror.development@gmail.com